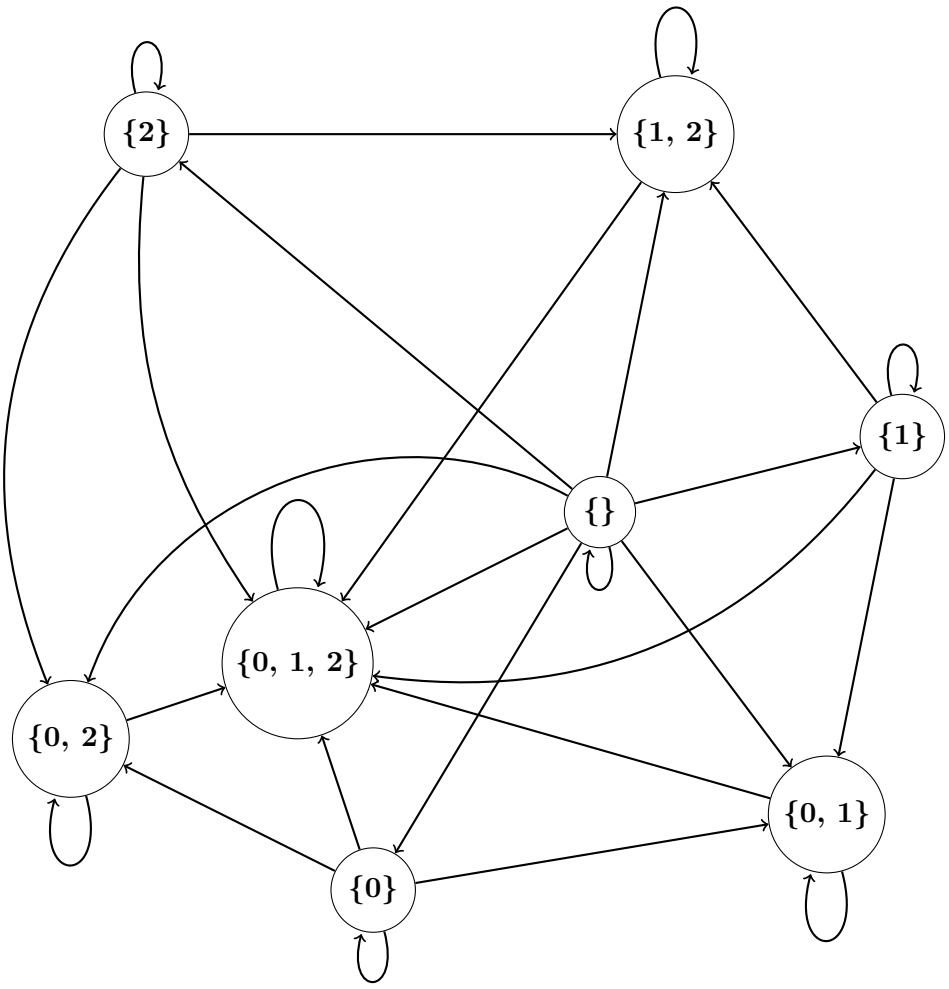
Student Information

Full Name : Ozan Akın
Id Number : 2309599

Answer 1

a)



b)

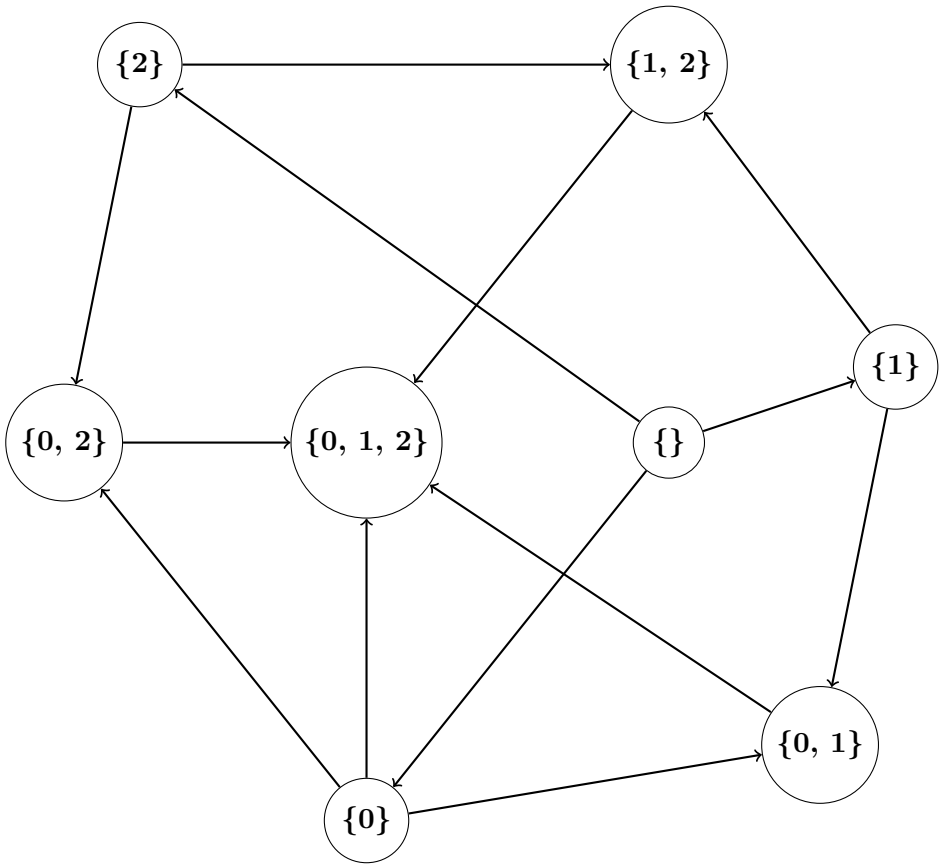
- Definition 1 from section 9.6 of the textbook says "A relation R on a set S is called a partial ordering or partial order if it is reflexive, antisymmetric, and transitive. A set S together with a partial ordering R is called a partially ordered set, or poset, and is denoted by (S, R). Members of S are called elements of the poset."
- Hence, we need to prove that graph is reflexive, antisymmetric, and transitive.
- Reflexivity: Since each element has a loop to itself, the graph is reflexive.
- Antisymmetry: Since there is no edge such that edge which is opposite direction of an edge of two vertices, the graph is antisymmetric.

- Transitivity: Since there are some vertices such that there is at least one edge from and to the graph, such as {1} and {2} the graph is transitive.
- Since the graph is reflexive, antisymmetric, and transitive, (S, R) is a poset.

c)

- Definition 3 from section 9.6 of the textbook says "If (S, R) is a poset and every two elements of S are comparable, S is called a totally ordered or linearly ordered set, and R is called a total order".
- Since (S, R) is a poset and we can compare two sets, (S, R) is a total order.

d)



e)

- Lattice definition from section 9.6 of the textbook says "A partially ordered set in which every pair of elements has both a least upper bound and a greatest lower bound is called a lattice.
- Since the upper bound of the vertices {1} and {2} are not the same, (S, R) is not a lattice.

Answer 2

a)

initial vertex	terminal vertices
a	
b	a, c
c	f
d	c, d, e, g
e	c, f, g
f	b
g	d

b)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

c)

vertex	in-degrees count	out-degrees count
a	2	0
b	1	2
c	3	1
d	2	5
e	1	3
f	2	1
g	2	1

d)

- $b \to c \to f \to b \to a$
- $d \to g \to d \to e \to c$
- $d \to c \to f \to b \to a$
- $e \to f \to b \to c \to f$
- $d \to e \to c \to f \to b$
- $e \to g \to d \to c \to f$

e)

- $b \to c \to f \to b$
- $c \to f \to b \to c$
- $d \to d \to g \to d$
- $d \to e \to g \to d$
- $d \to g \to d \to d$
- $e \to g \to d \to e$
- $f \to b \to c \to f$
- $g \to d \to d \to g$
- $g \to d \to e \to g$

f)

- Since there is no path from *a to b*, we can say that the graph is *not strongly connected*.
- By using Definition 5 from the textbook Section 10.4, we can say that if there is a path between every two vertices in underlying undirected graph, the directed graph is *weakly connected*.

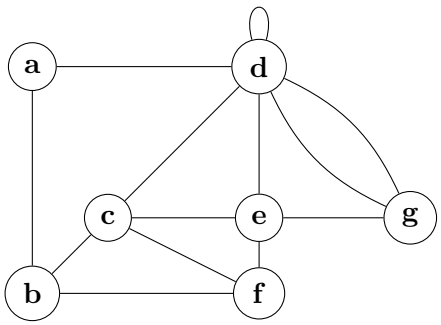


Figure 1: underlying undirected graph

- Since all vertices are connected a vertex except themselves, we can create a path between every two vertices.
- Hence, the direct graph is weakly connected.

g)

- the vertex a
- the subgraph containing vertices b, c, f, and edges $(b \to c)$, $(c \to f)$, $(f \to b)$
- the subgraph containing vertices d, e, g, and edges $(d \to e)$, $(e \to g)$, $(g \to d)$, $(d \to g)$

h)

- By using matrix from part b, we can generate adjacency matrix for subgraph H, the matrix A.

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- Since we want to find different paths from d to g with length 3, by using Theorem 2 from the section 10.4 of the textbook, we need A^3.

$$\begin{bmatrix} 4 & 2 & 1 & 3 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 2 \end{bmatrix}$$

- Hence, there are 3 different paths from d to g.

Answer 3

First, let's find degrees of vertices for the graph G.

a	b	c	d	e	f	g	h
2	3	2	5	4	2	2	2

a)

- Theorem 2 from the section 10.5 of the textbook says "A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree."
- Since, only the vertices `b and d` have odd degree, others have even degree, the graph G has an Euler path.

b)

- Theorem 1 from the section 10.5 of the textbook says "A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree."
- From the both part a and theorem 1, we can conclude that the graph G has not an Euler circuit since there are vertices with odd degree such as `b and d`.
- Hence, the graph G has not a Euler circuit.

c)

- Definition 2 from the section 10.5 of the textbook says "A simple path in a Graph G that passes through every vertex exactly once is called a Hamilton Path."
- Since we cannot create a path that passes through every vertex without passing the vertices `d and e` twice, we cannot create a path passes through every vertex only once.
- Hence, the graph G has not a Hamilton path.

d)

- Theorem 2 from the section 10.5 of the textbook (Dirac's Theorem) says "If G is a simple graph with n vertices with $n \geq 3$ such that the degree of every vertex in G is at least $n/2$, then G has a Hamilton circuit."
- Since the number vertices is **8**, the degree of every vertex in G must be $d \geq 4$.
- But the vertices `a, c, f, g and h` have degree of 2. Hence, the graph G has not a Hamilton circuit.

Answer 4

- To find whether these graphs are isomorphic, we can use adjacency matrices. If these matrices are identical, then the graphs are isomorphic.
- Adjacency matrix for graph G.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- Adjacency matrix for graph $G^{'}$.

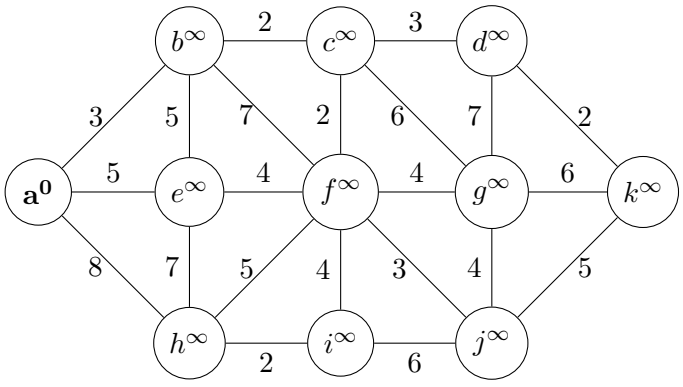
$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- Since these matrices are identical, these two graphs are isomorphic.

Answer 5

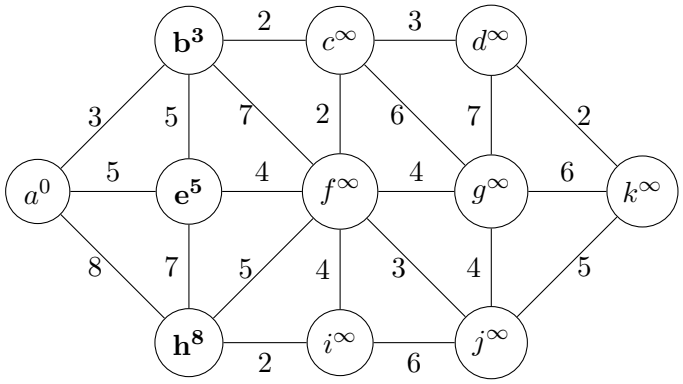
a)

- **Step 0**



- **Step 1**

  Visiting node a.

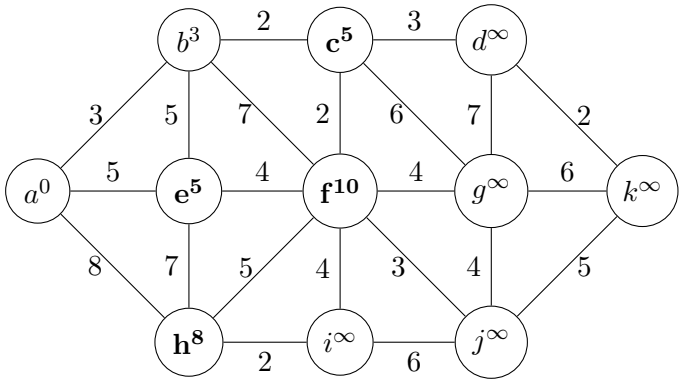


  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$

  Visited nodes: a

- **Step 2**

  Visiting node b.

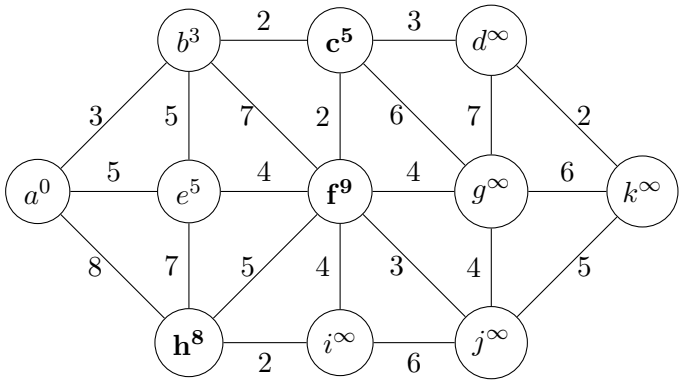


  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 10(a \to b \to f)$

  Visited nodes: a, b

- **Step 3**

  Visiting node e.

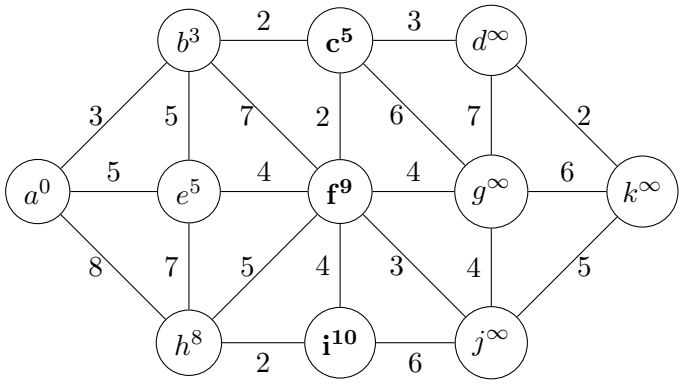


  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 9(a \to e \to f)$

  Visited nodes: a, b, e

- **Step 4**

  Visiting node h.

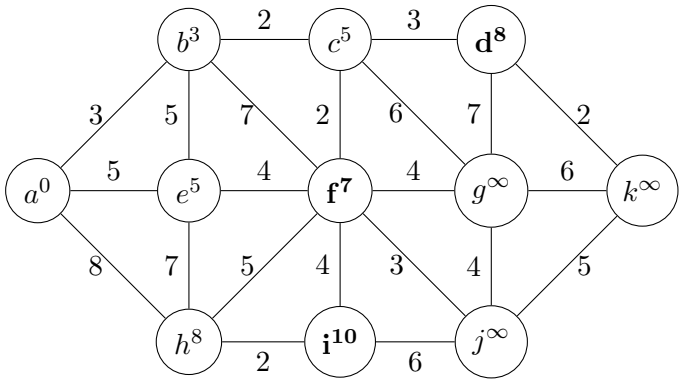


  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 9(a \to e \to f)$
  - $i : 10(a \to h \to i)$

  Visited nodes: a, b, e, h

- **Step 5**

  Visiting node c.

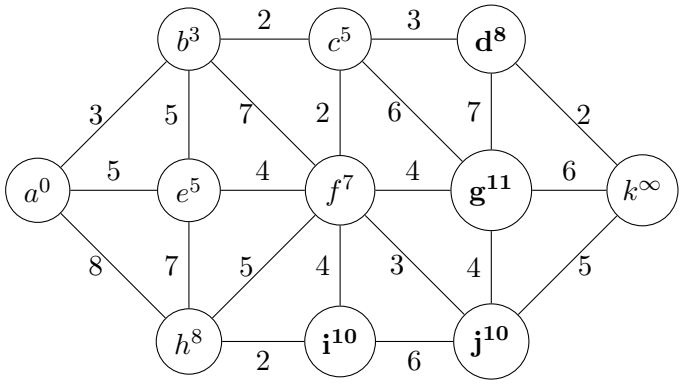


  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 7(a \to b \to c \to f)$
  - $i : 10(a \to h \to i)$
  - $d : 8(a \to b \to c \to d)$

  Visited nodes: a, b, e, h, c

- **Step 6**

  Visiting node f.

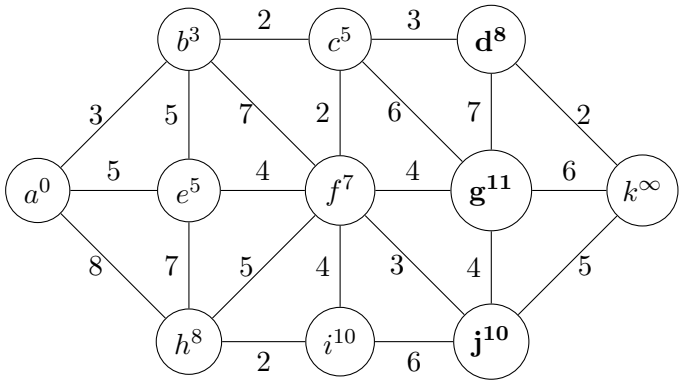


  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 7(a \to b \to c \to f)$
  - $i : 10(a \to h \to i)$
  - $d : 8(a \to b \to c \to d)$
  - $g : 11(a \to e \to f \to g)$
  - $j : 10(a \to e \to f \to j)$

  Visited nodes: a, b, e, h, c, f

- **Step 7**

  Visiting node i.

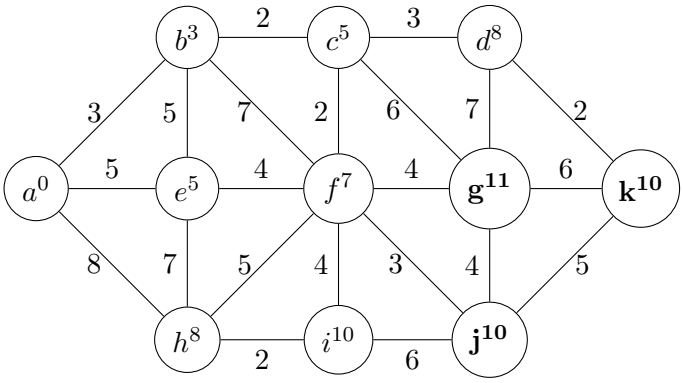


  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 7(a \to b \to c \to f)$
  - $i : 10(a \to h \to i)$
  - $d : 8(a \to b \to c \to d)$
  - $g : 11(a \to e \to f \to g)$
  - $j : 10(a \to e \to f \to j)$

  Visited nodes: a, b, e, h, c, f, i

- **Step 8**

  Visiting node d.



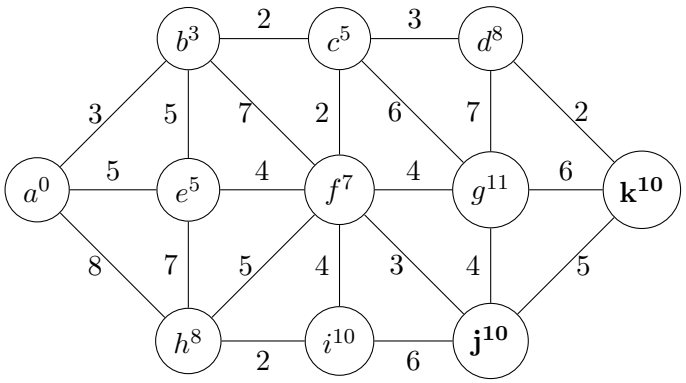
  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 7(a \to b \to c \to f)$
  - $i : 10(a \to h \to i)$

- $d : 8(a \to b \to c \to d)$
- $g : 11(a \to e \to f \to g)$
- $j : 10(a \to e \to f \to j)$
- $k : 10(a \to b \to c \to d \to k)$

Visited nodes: a, b, e, h, c, f, i, d

- **Step 9**

  Visiting node g.

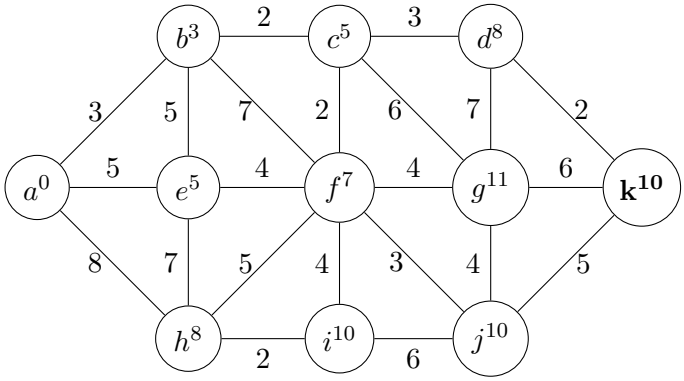


- $b : 3(a \to b)$
- $e : 5(a \to e)$
- $h : 8(a \to h)$
- $c : 5(a \to b \to c)$
- $f : 7(a \to b \to c \to f)$
- $i : 10(a \to h \to i)$
- $d : 8(a \to b \to c \to d)$
- $g : 11(a \to e \to f \to g)$
- $j : 10(a \to e \to f \to j)$
- $k : 10(a \to b \to c \to d \to k)$

Visited nodes: a, b, e, h, c, f, i, d, g

- **Step 10**

  Visiting node j.

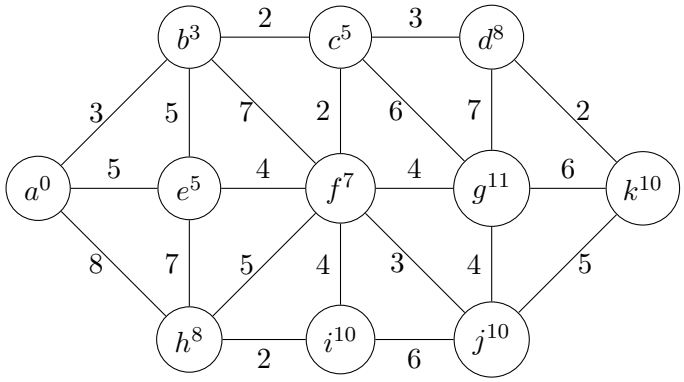


  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 7(a \to b \to c \to f)$
  - $i : 10(a \to h \to i)$
  - $d : 8(a \to b \to c \to d)$
  - $g : 11(a \to e \to f \to g)$
  - $j : 10(a \to e \to f \to j)$
  - $k : 10(a \to b \to c \to d \to k)$

  Visited nodes: a, b, e, h, c, f, i, d, g, j

- **Step 11**

  Visiting node k.



  - $b : 3(a \to b)$
  - $e : 5(a \to e)$
  - $h : 8(a \to h)$
  - $c : 5(a \to b \to c)$
  - $f : 7(a \to b \to c \to f)$

- $i : 10(a \to h \to i)$
- $d : 8(a \to b \to c \to d)$
- $g : 11(a \to e \to f \to g)$
- $j : 10(a \to e \to f \to j)$
- $k : 10(a \to b \to c \to d \to k)$

Visited nodes: a, b, e, h, c, f, i, d, g, j, k

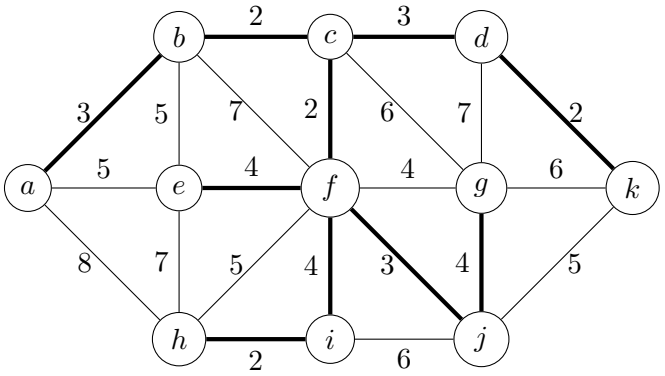
- **Step 12**

  Since there is no unvisited vertex left, the Djikstra algorithm has finished. The shortest path from a to j is the path $(a \to e \to f \to j)$ with the cost 10.

b)

Choice	Edge	Weight
1	{a,b}	3
2	{b,c}	2
3	{c,f}	2
4	{e,f}	4
5	{f,i}	4
6	{h,i}	2
7	{f,j}	3
8	{g,j}	4
9	{c,d}	3
10	{d,k}	2

Total: 29



Answer 6

a)

There are **7** vertices, **6** edges on T. It's height is **3**.

b)

```
a:17 b:13 c:24 d:19 e:43 f:23 g:58
```

c)

`b:13 d:19 f:23 g:58 e:43 c:24 a:17`

d)

`b:13 a:17 d:19 c:24 f:23 e:43 g:58`

e)

- Definition 3 from section 11.1 of the textbook says, "The tree is called a full m-ary tree if every internal vertex has exactly m children. An m-ary tree with m = 2 is called a binary tree."
- So in order to T be a full binary tree, every non-leaf node has exactly 2 children.
- Non-leaf nodes, namely `a, c, e`, has exactly 2 children. As a result, T is a full binary tree.

f)

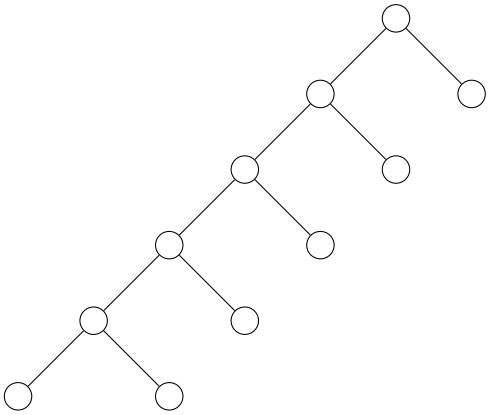
- Section 11.1 of the textbook says: "A complete m-ary tree is a full m-ary tree in which every leaf is at the same level."
- We know from the part e that the tree T is a full binary tree.
- In order to T be a complete binary tree, all leaves must be at same level. But leaves `d and f` are not in the same level. Hence, the tree T is not a complete binary tree.

g)

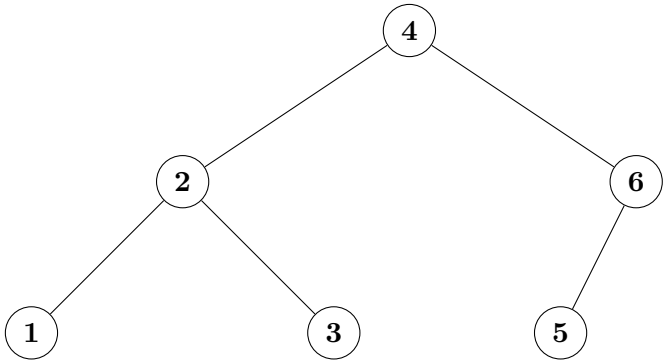
- In a binary search tree, if we do a inorder traversal the values of the vertices must be in ordered. (Since in inorder traversal, the order follows left-middle-right, and in a binary search tree middle values always greater than left values and always smaller than right values).
- From the part d, we can see that the vertices `c and f` are not in order. Hence, T is not a binary search tree.

h)

- In order to the graph be a full binary tree, all nodes except leaves must have exactly two children.
- For that purpose, for every non-leaf node must have one leaf and one non-leaf node except the nodes at level 5. This is the only way to both increase the height of the three same as following the rules for full-binary tree.
- So, in every level of the binary-tree there must be two nodes, except the root node. So we can create a formula for the minimum number of nodes in order to create a full binary tree as $node\_count = 2 \cdot h + 1$.
- Hence, the minimum number of nodes to create full binary tree with height 5 is, $2 * 5 + 1 = 11$.



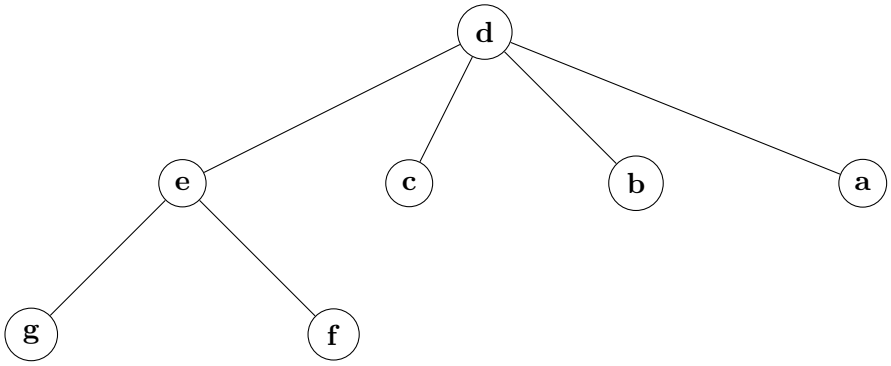
i)



j)

- Sequence to find 1: $4 \xrightarrow{\text{left}} 2 \xrightarrow{\text{left}} 1$
- Sequence to find 6: $4 \xrightarrow{\text{right}} 6$

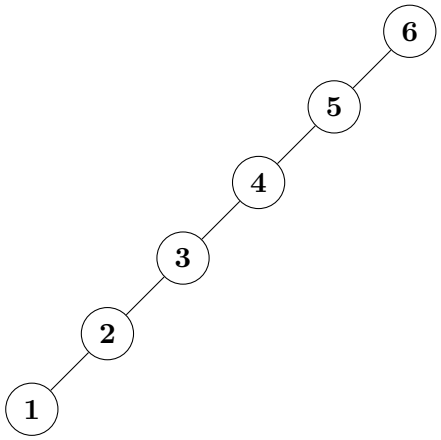
k)



l)

- If we want to have maximum height from the binary search tree, we can put only one vertex in every level of the binary tree. Using this method, we can generate tree with height $k - 1$ using k vertices.

- Let's take the set $S = \{1, 2, 3, 4, 5, 6\}$. If we want to create a binary search tree with maximum height, we can put these vertices like below.



- Since every level of binary search tree must be have at least one vertex, this is the only way to create binary search tree with maximum height. We cannot increase the height more without removing any of the level, which it is changes nothing if it removes a level from the binary search tree.
- Hence, using k vertex, binary search tree with maximum $k - 1$ height can be generated.